

The BQP-hardness of approximating the Jones Polynomial

Dorit Aharonov and Itai Arad*

Department of Computer Science and Engineering,
Hebrew University, Jerusalem, Israel

22-May-2006

Abstract

Following the work by Kitaev, Freedman and Wang [1], Aharonov, Jones and Landau [3] recently gave an explicit and efficient quantum algorithm for approximating the Jones polynomial of the plat closure of a braid, at the k th root of unity, for constant k . The universality proof of Freedman, Larsen and Wang [2] implies that the problem which these algorithms solve is BQP-hard. The fact that this is the only non-trivial BQP-complete problem known today motivates a deep investigation of this topic.

A natural question which was raised in [3] is the following. The results of [3] actually gave efficient algorithms also in the case of asymptotically growing k 's - up to k which is polynomial in the size of the braid. However, the results of [2] only imply universality in the case of constant k , via the Solovay-Kitaev theorem; The application of this theorem relies heavily on the fact that the generators of the groups in question are fixed. The question of the complexity of the problems with asymptotically growing k was thus left open.

In this paper we resolve this question and prove that the Jones polynomial approximation problem is BQP-complete also for asymptotically growing k (bounded by a polynomial). To do this we introduce some new techniques for analyzing universality in quantum computation, which enable us to apply Solovay-Kitaev indirectly. As a side benefit, we reprove the density theorem of [2] using quite elementary arguments; this hopefully sheds light on the reason that these problems are indeed quantum-hard.

*email: itaia@cs.huji.ac.il

1 Introduction

What is the computational power of quantum computers? This question is fundamental both from a computer scientist as well as a physicist points of view. This paper attempts to improve our understanding of this question, by studying the only non-trivial BQP-complete problem known to us today: the problem of approximating the Jones polynomial.

The Jones polynomial, discovered in 1985 [4], is a very important knot invariant in topology. Its importance was manifested in connections to numerous areas in mathematics, from the statistical physics model known as the Potts model to the study of DNA folding. Among its many connections, an extremely important one was drawn by Witten in 1989 [5], to quantum mechanics; more precisely, to Topological Quantum Field Theory (TQFT). Witten showed how the Jones Polynomial and other topological invariants of links naturally appear in the Wilson lines of the $SU(2)$ Chern-Simons Topological Quantum Field Theory.

About a decade later, TQFT entered the scene of quantum computation, when Freedman suggested a computational model based on this theory [6]. The works of Freedman, Kitaev, Larsen and Wang [2, 1, 7] showed an equivalence between the TQFT model and the standard model of quantum computation. On one hand, they gave an efficient simulation of TQFT by a quantum computer [1]. On the other hand, they showed that quantum computation can simulate TQFT efficiently [2]. These results draw interesting links between quantum computation and the Jones polynomial. The simulation of TQFT by quantum computers implicitly implies the existence of a quantum algorithm for the Jones polynomial approximation problem at the fifth root of unity $e^{i2\pi/k}$, via the results of Witten; The simulation of quantum computers by TQFT implicitly implies that the Jones polynomial approximation problem is BQP-hard. However, these important results were stated in TQFT language, and were not stated or proved explicitly in the above set of works. A clear statement of some of the algorithmic result, in a computational language, was given in [8], but without an explicit algorithm.

Only recently Aharonov, Jones and Landau [3] provided an explicit and efficient quantum algorithm for the problem of approximating the Jones polynomial at those roots of unity. The algorithm uses a combination of mathematical results of over 20 years ago due to Jones. In particular, the main ingredient of the algorithm is a unitary representation of the braid group called the *path model representation*, in which braids are mapped to operators acting on a Hilbert space, whose basis vectors are indexed by paths on a certain graph. This gave a simple to state algorithm for the problem, bypassing the TQFT language altogether. In the other direction, the universality proof due to [2] can also be made explicit in the standard quantum model language, without referring to TQFT. To do this one needs to encode the space of n qubits into the space of paths; a simple way to do it, encoding one qubit in four steps of a

path, was independently discovered by Kitaev [9], and Wocjan and Yard [10]. The latter also defined a universal set of gates that acts on these paths, and proved that certain additive approximations of the Jones polynomial are BQP-complete.

The above two results together give an explicit proof that the problem of approximating the Jones polynomial at the fifth root of unity, and in fact, for any primitive root of unity $e^{2\pi i/k}$, for constant $k > 4, k \neq 6, 10$ is BQP-complete. This is the only non-trivial BQP-complete problem known today¹. This fact highlights the importance of this problem in the context of the quantum computational power, and motivates deeper investigation of this topic. One natural direction to try to take is generalizations of the algorithms. Initial steps in this direction were made in [11, 10]. Another interesting direction is to study the reasons for BQP-hardness, and its range of applicability. This is the path we take in this paper.

One natural question is the following. It turns out that the algorithms given in the work of Aharonov *et. al* work not only for constant k , but also for asymptotically growing k 's. To be more precise, [3] gives an efficient quantum algorithm to approximate the Jones polynomial of an n strands braid with m crossings at a primitive root of unity $e^{2\pi i/k}$, where the running time of the algorithm is polynomial in m, n and k . The algorithm is therefore efficient even if k grows polynomially with n . On the other hand, we only know that the constant k case is BQP-hard. Therefore, in [3] the following natural question was raised: what is the complexity of approximating the Jones polynomial for polynomially bounded k ? It was left open whether it is BQP-hard, doable in BPP, or maybe somewhere in between.

In this paper we resolve this question, and show that it is also BQP-hard:

Theorem 1.1 *The problem of approximating the Jones polynomial of the plat closure of a given braid b , with m crossings, at $e^{2\pi i/k}$, where both m and k are polynomially bounded in n , (and k is some fixed function of n), is BQP-complete.*

We outline the difficulties in the proof, and our methods to overcome them, below. We hope that this paper makes a significant step towards better understanding of the complexity of the Jones polynomial approximation problems, and of quantum computational complexity in general. In particular, we hope that the techniques we developed here would help in proving BQP-hardness of future quantum algorithms to follow.

¹In this paper we use the term BQP-complete loosely - in fact, this is a promise problem, and therefore is not complete in the usual sense.

1.1 Proof Outline

Given an algorithm that calculates the Jones Polynomial of any link at $e^{-2\pi i/k}$ (for some integer $k > 4$ and $k \neq 6, 10$) in polynomial time in the number of crossings in the link, and a classical Turing machine - we can simulate a Quantum computer efficiently. How is that possible? The key idea here, which is demonstrated in [3], is to use an intimate connection between two, seemingly distinct, worlds: links and unitary matrices. The connection is the so-called “path representation” which is a unitary representation of the braid group. For a fixed k , the k th path model representation of the braid group maps every braid to a unitary operator on a Hilbert space spanned by paths on a certain graph which depends on k . In particular, each generator of the braid group (which is simply one crossings of two adjacent strands) is mapped to a certain (k -dependent) unitary operator on the space of paths. The BQP-hardness proof boils down to showing that a general quantum gate can be approximated efficiently using these unitary operators.

Let us first review the easier constant k case. Here, the main difficulty in the proof is to show that the relevant set of operators is dense in some subgroup of the unitary group. Once this is shown, it is standard to apply the famous Solovay-Kitaev theorem to show that *density implies efficiency*. In other words, once the subgroup is dense, then Solovay-Kitaev give a method to approximate every gate in the quantum circuit by a short sequence of generators, and universality follows.

But how does one prove the density? The starting point is the fact that Kitaev’s four steps encoding encodes two-qubit gates into the unitary group operating on the space spanned by 14 paths. Density thus means that we can approximate any matrix in $SU(14)$, using our k -generators. The idea here is to first restrict attention to some two dimensional subspace, and show density in $SU(2)$. This was essentially done by Jones [12]. We then gradually increase the dimensionality of the space on which we have density, to $SU(14)$, by adding one or more dimensions at a time; to this end we introduce two general and very useful lemmas: the bridge lemma 4.1 and the decoupling lemma 4.2. This completes the density proof of the constant k case; we get an almost self-contained, fairly elementary proof.

We would now like to move to the asymptotically growing k case. Here, however, there is a subtle point in the above line of arguments. Indeed, density still holds. But the step of *density implies efficiency* fails. The starting point of the Solovay-Kitaev theorem is the construction, using the set of generators, of an epsilon-net in the unitary group, where epsilon is some small enough constant. Such an epsilon-net is easy to construct, given a *fixed* set of generators that span a dense subgroup - essentially, brute force would do the trick. However, if k is asymptotically growing in n , the set of generators is no longer fixed. It is no longer clear that the very first step of the Solovay-Kitaev theorem, that of creating the epsilon-net, can be done efficiently.

We give here a very rough sketch of how we overcome this difficulty. We first choose a large enough but fixed k_0 . We now modify the k_0 -generators as follows: we leave their eigenvalues as they were, but change their basis to the basis of eigenvectors of the generators corresponding to the case $k \mapsto \infty$. This change of basis turns out not to affect the fact that the set of generators generates a dense subgroup. Since the new set of generators is fixed (independent of n), we can find an epsilon-net which consists of these modified k_0 -generators simply by brute-force. The advantage of using the modified generators as the building blocks of our epsilon-net is that the eigenvectors of the modified generators are very close to the eigenvectors of generators corresponding to very large k 's. If we want to create an epsilon-net using k -generators for very large k 's, we only need to approximate the modified generators, using these k -generators. Since the eigenvectors more or less agree, we only need to approximate the eigenvalues, which can be done by taking the appropriate power of the relevant k -generator. We get an efficient construction of an epsilon-net consisting of k -generators. We can now apply the Solovay-Kitaev theorem using this net.

We now proceed to the detailed proof, but before this, we first need to review some basic facts about the Braid group, and introduce the path model representation.

2 Mathematical Background

2.1 The braid group B_n

Loosely speaking, a braid is a set of n strands that connect two horizontal bars, such that each strand is tied exactly to one peg on the top bar and one peg on the bottom bar. When drawing the braid schematically on a paper, the strands may pass over and under each other, but at any point they must not be completely horizontal. Braids which can be deformed into each other without tearing any of the strands are considered identical. An illustration of a 4-strand braid is given in Fig. 1.

The set of all braids with n strands forms an infinite and discrete group which is called the *braid group* B_n . The product rule for $b_1 b_2$ is defined by placing the braid b_1 above the braid b_2 and fusing the bottom of the b_1 strands with the top of the b_2 strands. The identity element is the braid with n straight lines that connect each peg at the bottom bar to its corresponding peg at the upper bar.

In [13], Artin proved that B_n admits a finite presentation (the *Artin presentation*) with $n-1$ generators $\{\sigma_i\}$ that satisfy the following constraints:

$$\sigma_i \sigma_j = \sigma_j \sigma_i \quad \text{for } |i - j| \geq 2, \quad (1)$$

$$\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}. \quad (2)$$

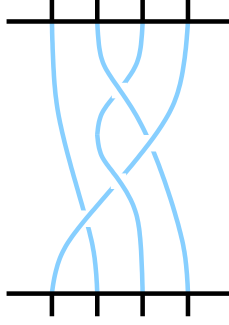


Figure 1: An example of a 4-strand braid

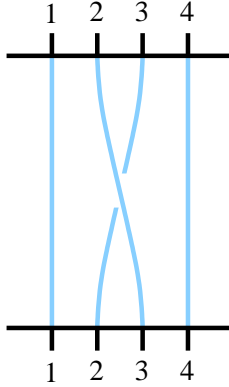


Figure 2: The generator σ_2 in the braid group B_4 .

Pictorially, σ_i is a braid that is identical to the unity braid in all strands except for the i and $i + 1$ strands which cross each other once (the $i + 1 \rightarrow i$ strand goes over the $i \rightarrow i + 1$ strand), connecting the lower i 'th peg to the upper $i + 1$ peg and vice versa. An illustration of σ_2 is given in Fig. 2. It is an easy exercise to verify graphically that the braid generators indeed satisfy (1), (2).

2.2 The path representation

The path representation is a unitary representation of the braid group B_n that depends on an integer $k \geq 3$. The image of every $b \in B_n$ under this representation is denoted by $\rho(b)$ and it acts on a finite Hilbert space. To understand the structure of this space, we introduce the graph G_k which is a set of $k - 1$ sites (vertices) and $k - 2$ edges that connect them. The sites are ordered from bottom to top one above the other, as described in Fig. 3. To each site we assign a number according to its position; the number 1 is assigned to the bottom site while the number $k - 1$ is assigned to the top site. We then consider all possible n -steps walks (paths) over the graph G_k which start at site 1 *and never leave* G_k . With each

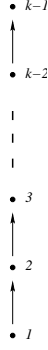


Figure 3: The graph G_k

path we associate a vector from the standard basis of \mathcal{B}^n by mapping the qubit 1 with a movement in the upper direction (i.e., moving from site i to site $i + 1$), and the qubit 0 with a movement in the opposite direction. For example, the vector $|1011\rangle$ represents the 4-steps walk $1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 3$. Of course, not all standard basis elements correspond to valid paths. For example, since all paths begin at the bottom site and we are not allowed to leave the graph, they must start with the bit 1. The subspace of all the legitimate n -steps paths is denoted by $H_{n,k}$.

To define ρ , it is enough to specify its action on the braid group generators σ_i , which we denote by $\rho_i \stackrel{\text{def}}{=} \rho(\sigma_i)$. We describe ρ_i in terms of *partial paths*: if p is a bit string that corresponds to a path, then $p|_i$ denotes the first $i - 1$ bits of the string and $p|^i$ denotes the last bits of the string starting from the $i + 2$ bit. For example, if $p = \text{"11100101"}$, then $p|_3 = \text{"11"}$ and $p|^3 = \text{"0101"}$, hence $p = p|_3 10 p|^3$. Next, we let z_i denote the position on the graph after the path $p|_i$ was taken. So for $p = \text{"11100101"}$, we have $z_1 = 1, z_2 = 2, z_3 = 3, z_4 = 4, z_5 = 3$ etc. Finally, we define

$$\theta \stackrel{\text{def}}{=} \pi/k, \quad (3)$$

$$\lambda_\ell \stackrel{\text{def}}{=} \sin(\pi\ell/k) = \sin(\ell\theta), \quad (4)$$

$$A \stackrel{\text{def}}{=} ie^{-i\pi/2k} = ie^{-i\theta/2}. \quad (5)$$

Notice that definition (4) is for $\ell = 1, \dots, k - 1$. For $\ell < 1$ or $\ell > k - 1$, we set $\lambda_\ell = 0$.

Using this notation we define the operators Φ_i with $i = 1, \dots, n - 1$: for every initial/final strings p_i ,

p^i , we have

$$\Phi_i |p|_i 00 p|^i\rangle = 0 , \quad (6)$$

$$\Phi_i |p|_i 01 p|^i\rangle = \frac{\lambda_{z_i}-1}{\lambda_{z_i}} |p|_i 01 p|^i\rangle + \frac{\sqrt{\lambda_{z_i}+1}\lambda_{z_i}-1}{\lambda_{z_i}} |p|_i 10 p|^i\rangle , \quad (7)$$

$$\Phi_i |p|_i 10 p|^i\rangle = \frac{\lambda_{z_i}+1}{\lambda_{z_i}} |p|_i 10 p|^i\rangle + \frac{\sqrt{\lambda_{z_i}+1}\lambda_{z_i}-1}{\lambda_{z_i}} |p|_i 01 p|^i\rangle , \quad (8)$$

$$\Phi_i |p|_i 11 p|^i\rangle = 0 . \quad (9)$$

Then ρ_i are given by

$$\rho_i \stackrel{\text{def}}{=} A\Phi_i + A^{-1}\mathbb{1} = A^{-1}(A^2\Phi_i + \mathbb{1}) = A^{-1}(\mathbb{1} - e^{-i\theta}\Phi_i) . \quad (10)$$

In [3] it is shown that that these operators indeed form a unitary representation of the braid group.

Notice that the operators ρ_i are invariant under the same subspaces of the Φ_i operators. Specifically, they multiply the vectors $|p|_i 00 p|^i\rangle$, $|p|_i 11 p|^i\rangle$ by the phase A^{-1} , while mixing the vectors $|p|_i 10 p|^i\rangle$, $|p|_i 01 p|^i\rangle$. Therefore in the standard basis, ρ_i breaks into one-dimensional and two-dimensional blocks (but notice that these are different blocks for different operators). The two-dimensional blocks mix the $|p|_i 10 p|^i\rangle$, $|p|_i 01 p|^i\rangle$ vectors whose corresponding paths end at the same point. It follows that if $|p\rangle$ corresponds to a path that ends at ℓ , then $\rho_i|p\rangle$ can be written as a linear combinations of vectors that correspond to paths that end at the site ℓ . This is true for *all* ρ_i , hence the path representation breaks into representations over subspaces that correspond to paths that end at a particular ℓ . We denote these subspaces by $H_{n,k,\ell}$.

2.3 From braids to links to Jones Polynomial

As stated in the beginning of this section, the path representation provides a connection between the Jones polynomial of links to unitary matrices. We will now explore this connection in some depth. In particular, we will see that quantum mechanical expressions of the form $\langle 0|U|0\rangle$, which are sufficient to describe quantum computation, can be given in terms of the Jones polynomial.

We first notice that a braid can be transformed into a link by connecting its open endpoints. Such an operation is called a *closure*, and here we focus on one particular closure: the *plat closure*. This closure is defined only for braids with an even number of strands. It is the link that is formed by connecting the top pegs with odd numbers with the peg to their right, and doing the same with the bottom pegs. The plat closure of a braid $b \in B_n$ is denoted by b^{pl} . Figure 4 shows the plat closure of the 4-strand braid from Fig. 1. It turns out that there is a very strong connection between the path representation of a braid and the Jones polynomial of its plat closure.

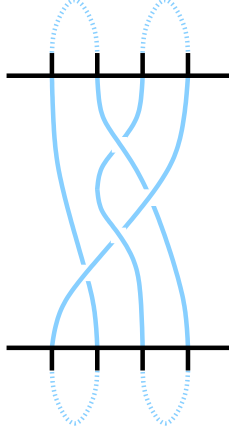


Figure 4: The plat closure of the 4-strand braid from Fig. 1

In [3] it was shown that the Jones Polynomial of the plat closure of every $b \in B_n$ can be given by a “sandwich” product of the operator $\rho(b)$ with a special vector $|\alpha\rangle \in H_{n,k}$. Specifically, let $V_{b^{pl}}(\cdot)$ denote the Jones polynomial of b^{pl} , and $|\alpha\rangle$ be the “zig-zag” vector

$$|\alpha\rangle = |10101 \dots 10\rangle, \quad (11)$$

that corresponds to a path in which the first step is up, the second is down, the third is up etc. Then according to [3], the following equality holds:

$$\langle \alpha | \rho(b) | \alpha \rangle = \frac{N}{\lambda_1} d^{-(n-1)} (-A)^{-3w(b^{pl})} V_{b^{pl}}(A^{-4}). \quad (12)$$

$w(b^{pl})$ is the *writhe* of the link b^{pl} , and $V_{b^{pl}}(A^{-4})$ is the Jones polynomial of b^{pl} , evaluated at $A^{-4} = e^{-2\pi i/k}$. The exact definition of these two quantities is given in [14, 3]. The other quantities in the formula are λ_ℓ and A which are defined in Eqs. (4, 5) respectively, and N and d which are defined by

$$N \stackrel{\text{def}}{=} \sum_{\ell=1}^{k-1} \lambda_\ell \dim(H_{n,k,\ell}), \quad (13)$$

$$d \stackrel{\text{def}}{=} -A^2 - A^{-2}. \quad (14)$$

Note that the writhe is a trivial function of a link, as it is basically a sum over all its crossings. The Jones Polynomial, on the other hand, is a very non-trivial function of the link; the best known classical algorithms for its calculation are exponential in the number of crossings.

Notice also that these two quantities are only defined for *oriented* links, and therefore we must choose some orientation for b^{pl} to calculate them. It does not matter, however, which orientation we pick since the combination $(-A)^{-3w(b^{pl})} V_{b^{pl}}(A^{-4})$ is independent of the orientation (in agreement with the LHS of

the equation). It is actually the *Kauffman bracket* $\langle b^{pl} \rangle(A)$ of the link b^{pl} - which is also a polynomial of the link. For further details we refer the reader to [14].

3 Simulating a quantum computer

With the path representation at hand, together with Eq. (12), we can devise a plan to simulate a quantum computer. The idea is straightforward: given a description of a quantum circuit U over n qubits, which is a $poly(n)$ sequence of unitary gates from a universal basis, we find a braid b such that

$$\langle \alpha | \rho(b) | \alpha \rangle \simeq \langle 0 | U | 0 \rangle . \quad (15)$$

Here the \simeq sign means that we can approximate the RHS by the LHS to any desired accuracy. Thus if we have an oracle that, given b , tells us the Jones polynomial $V_{b^{pl}}(e^{-2\pi i/k})$, then from Eq. (12) we can easily find the LHS of Eq. (15), which approximates $\langle 0 | U | 0 \rangle$. This is sufficient to simulate quantum computation because of the following reasoning. Suppose we want to know the probability that a given quantum circuit Q outputs 0 in its last qubit when applied on an input string X and then measured. We replace Q by a circuit U with one more qubit, which starts by applying *NOT* gates on the qubits that are 1 in the input string X , then applies Q , then copies the last qubit of Q to the additional qubit by a *CNOT* gate, and then applies Q in reverse, and also reverses the initialization of the input qubits to get 0. It is a simple algebra exercise to show that $\langle 0 | U | 0 \rangle$ is equal to the probability to get 0 in the original circuit Q applied to X .

The algorithm for finding the braid b , must be polynomial in n (the number of qubits in U) if we want our simulation to be polynomial. Additionally, if we assume that we can calculate the Jones polynomial in a time which is polynomial in the number of its crossings, then we must also assure that the number of crossings in b is polynomial in n , in order to keep the overall simulation polynomial.

3.1 The 4-steps encoding

We would now like to find a braid b that corresponds to the operator U as in Eq. (15). We first simplify the matrices we deal with. It is easy to verify that any quantum circuit can be translated, with a linear overhead, to an equivalent quantum circuit which acts on qubits arranged in a one-dimensional array, and applies gates only on nearest neighbor qubits. We therefore assume that U is given by a product of $poly(n)$ basis elements $U = U_L \cdot \dots \cdot U_1$, ($L = poly(n)$) each of them acting non-trivially only on two adjacent qubits.

The next step is to encode U in path space as it is the space on which the operators $\rho(b)$ act. One

such convenient encoding represents every qubit by a 4-steps path:

$$|\underline{0}\rangle \stackrel{\text{def}}{=} |1010\rangle \quad (16)$$

$$|\underline{1}\rangle \stackrel{\text{def}}{=} |1100\rangle . \quad (17)$$

Then a string of n encoded qubits is encoded as a $4n$ -steps path in $H_{4n,k}$, and every gate U is encoded as a $2^{4n} \times 2^{4n}$ matrix \underline{U} :

$$\underline{U} = \sum_{i,j} U_{ij} |\underline{i}\rangle \langle \underline{j}| + \mathbb{1}_{\text{over rest of space}} . \quad (18)$$

Here $|\underline{i}\rangle$ denotes the $4n$ -steps path that encodes the qubits in the original vector $|i\rangle$. Notice, however, that this is not an arbitrary path in $H_{4n,k}$ as it returns to the initial site every 4 steps. We denote the subspace that is spanned by all these paths by S .

Next, the product $U = U_L \cdot \dots \cdot U_1$ naturally translates to $\underline{U} = \underline{U}_L \cdot \dots \cdot \underline{U}_1$ and so by finding braids $b_i \in B_{4n}$ such that $\rho(b_i) \simeq \underline{U}_i$ and then taking their product $b = b_L \cdot \dots \cdot b_1$, we will get $\rho(b) \simeq \underline{U}$. Consequently we would have:

$$\langle 0^{\otimes n} | U | 0^{\otimes n} \rangle = \langle \underline{0}^{\otimes n} | \underline{U} | \underline{0}^{\otimes n} \rangle \simeq \langle \underline{0}^{\otimes n} | \rho(b) | \underline{0}^{\otimes n} \rangle = \langle \alpha | \rho(b) | \alpha \rangle . \quad (19)$$

In fact, we will not be so ambitious; we will only require that $\rho(b_i) \simeq \underline{U}_i$ on the subspace S , and show that this suffices.

The advantage of using this particular encoding is that, together with the tensorial structure of the qubits, it allows us to concentrate on the “reduced” braid group B_8 instead of the larger group B_{4n} . Let us explain exactly what is meant by that. Suppose that we wish to perform an operation on the $s, s+1$ encoded qubits of some path $|p\rangle \in S$. Then we must use a braid $b \in B_{4n}$ that mixes the 8 strands $4(s-1)+1 \rightarrow 4(s+1)$ while being trivial on the rest. However, since $|p\rangle \in S$, its path reaches the first site before the $4(s-1)+1$ and $4(s+1)+1$ steps. Therefore the three partial paths that are defined by the steps $1 \rightarrow 4(s-1)$, $4(s-1)+1 \rightarrow 4(s+1)$ and $4(s+1)+1 \rightarrow 4n$ are all *legitimate* paths over the graph G_k (i.e., they start and end at the first site and never leave G_k). We denote these partial paths by p_0, \tilde{p} and p_1 respectively, and write $|p\rangle = |p_0\rangle \otimes |\tilde{p}\rangle \otimes |p_1\rangle$. Notice also that $|\tilde{p}\rangle \in H_{8,k,1}$. We will add a tilde to all vectors and operators that act on that space. In particular, we define $\tilde{b} \in B_8$ to be the “reduced” version of b , created by the 8 non-trivial strands of b .

It is now easy to verify that

$$\rho(b)|p\rangle = \rho(b)\left(|p_0\rangle \otimes |\tilde{p}\rangle \otimes |p_1\rangle\right) = |p_0\rangle \otimes \left(\rho(\tilde{b})|\tilde{p}\rangle\right) \otimes |p_1\rangle . \quad (20)$$

This follows from the definition of the generators ρ_i in Eqs. (6-9, 10) which only depend on z_i - the position of the path after $i-1$ steps, and not on the index i itself.

By linearity, we can extend Eq. (20) to all vectors in S , which are simply superpositions of encoded paths. Therefore, as long as $|p\rangle \in S$, it is enough to search for an appropriate braid in the much simpler group, B_8 , instead of looking in the full B_{4n} group. What remains to show is that (i) we can approximate any operator on $H_{8,k,1}$ using a $\tilde{b} \in B_8$ (and that this can be done efficiently), and (ii) that the state we work with is always sufficiently close to the subspace S where Eq. (20) is valid. The next theorem and its subsequent claim show exactly that.

Theorem 3.1 (Density and efficiency in B_8) *Let \tilde{U} be an encoded two-qubit quantum gate, and let $\delta > 0$. Then there exists a braid $\tilde{b} \in B_8$, consisting of $\text{poly}(k, 1/\delta)$ generators of B_8 , such that*

$$\|(\rho(\tilde{b}) - \tilde{U})|\tilde{p}\rangle\| \leq \delta ,$$

for every $|\tilde{p}\rangle \in H_{8,k,1}$, provided that $k > 4$, $k \neq 6, 10$. Moreover, \tilde{b} can be found in $\text{poly}(k, 1/\delta)$ time.

We will devote most of the remainder of the paper to proving this theorem. Before we start, however, let us see how, together with Eq. (20), it can be used to construct the appropriate braid $b \in B_{4n}$ in Eq. (19).

Let $U = U_L \cdots U_1$ with U_i being local two-qubit gates, and let $\epsilon > 0$ be an arbitrarily constant. For every U_i we use the theorem to construct a braid $\tilde{b}_i \in B_8$, with $\delta = \epsilon/L$, and extend it into a braid $b_i \in B_n$ by adding identity strands at the appropriate places. Finally, b is taken to be the product of these b_i 's. We have

Claim 3.1 $\|\underline{U}_L \cdots \underline{U}_{L-1} \cdots \underline{U}_1 |\alpha\rangle - \rho(b_L) \cdots \rho(b_{L-1}) \cdots \rho(b_1) |\alpha\rangle\| \leq \epsilon$.

Proof: The claim is easily proved by induction. Indeed, assume that

$$\|\underline{U}_{i-1} \cdots \underline{U}_1 |\alpha\rangle - \rho(b_{i-1}) \cdots \rho(b_1) |\alpha\rangle\| \leq \frac{i-1}{L} \epsilon , \quad (21)$$

and define $|\beta\rangle \stackrel{\text{def}}{=} \underline{U}_{i-1} \cdots \underline{U}_1 |\alpha\rangle$. It is easy to verify that *any* encoded gate \underline{U} sends the subspace S into itself and therefore $|\beta\rangle \in S$. Consequently

$$\|\underline{U}_i |\beta\rangle - \rho(b_i) |\beta\rangle\| = \|\tilde{U}_i |\tilde{\beta}\rangle - \rho(\tilde{b}_i) |\tilde{\beta}\rangle\| \leq \frac{1}{L} \epsilon , \quad (22)$$

where the first equality follows from the reduction in Eq. (20) and the second inequality follows from the way in which we constructed $\tilde{b}_i \in B_8$. Then using the induction assumption together with the triangle inequality, we get

$$\begin{aligned} & \|\underline{U}_i \cdots \underline{U}_1 |\alpha\rangle - \rho(b_i) \cdots \rho(b_1) |\alpha\rangle\| \\ &= \|\underline{U}_i |\beta\rangle - \rho(b_i) |\beta\rangle + \rho(b_i) \underline{U}_{i-1} \cdots \underline{U}_1 |\alpha\rangle - \rho(b_i) \cdots \rho(b_1) |\alpha\rangle\| \\ &\leq \|\underline{U}_i |\beta\rangle - \rho(b_i) |\beta\rangle\| + \|\underline{U}_{i-1} \cdots \underline{U}_1 |\alpha\rangle - \rho(b_{i-1}) \cdots \rho(b_1) |\alpha\rangle\| \\ &\leq \frac{\epsilon}{L} + (i-1) \frac{\epsilon}{L} = i \frac{\epsilon}{L} . \end{aligned} \quad (23)$$

□

This shows that the braid $b = b_L \cdots b_1$ satisfies $\langle \alpha | \rho(b) | \alpha \rangle \simeq \langle \alpha | U | \alpha \rangle$ as required by Eq. (15) and Eq. (19). By theorem 3.1, the number of braid generators that are needed to approximate every gate is of the order $\text{poly}(k, 1/\delta)$, and they can be found in time $\text{poly}(k, 1/\delta)$. Since we have $L = \text{poly}(n)$ gates then the total number of generators is $\text{poly}(n, k, 1/\delta)$. Finally, as we assume that $k = \text{poly}(n)$ then the total number of generators is $\text{poly}(n, 1/\delta)$. This is exactly the number of crossings in the resultant link since every generator contains exactly one crossing. We have thus proved Theorem 1.1, which states that approximating the Jones polynomial of a plat closure of a braid with m crossings in $e^{2\pi i/k}$, with both m and k polynomially bounded in n - is BQP-complete.

In the next section we will prove theorem 3.1 - the B_8 density and efficiency theorem. We first show that the images of B_8 under the path representation form a dense subset of $SU(H_{8,k,1})$. This, together with the Solovay-Kitaev theorem would prove theorem 3.1 *for constant k*. We complete the proof in Sec. 4.3, where we extend the theorem to the general, $k = \text{poly}(n)$, case.

4 Proving the B_8 density and efficiency theorem

To prove theorem 3.1 let us first analyze the structure of the subspace $H_{8,k,1}$ and the generators ρ_1, \dots, ρ_7 of the B_8 path representation that act on it.

4.1 The structure of the generators in $H_{8,k,1}$

In Sec. 2.2 we saw that the generators break into 2-dimensional and 1-dimensional blocks when represented in the standard basis. Let us look at these blocks in some more detail.

Consider first the action of ρ_i on vectors with $z_i = 1$: here $\lambda_{z_i-1} = 0$ and consequently $|p|_i 10 p|^i\rangle$ is an eigenvector of Φ_i :

$$\Phi_i |p|_i 10 p|^i\rangle = \frac{\lambda_2}{\lambda_1} |p|_i 10 p|^i\rangle = 2 \cos \theta |p|_i 10 p|^i\rangle . \quad (24)$$

It is therefore an eigenvector of ρ_i with eigenvalue

$$A^{-1}(1 - 2e^{-i\theta} \cos \theta) = -A^{-1}e^{-i2\theta} . \quad (25)$$

The vectors $|p|_i 01 p|^i\rangle, |p|_i 00 p|^i\rangle$ do not exist since they represent a path that leaves the graph. Finally the vector $|p|_i 11 p|^i\rangle$ is nullified by Φ_i and is therefore also an eigenvector of ρ_i with the eigenvalue A^{-1} .

Next, consider the $z_i > 1$ case. Here Φ_i nullifies the $|p|_i 00 p|^i\rangle, |p|_i 11 p|^i\rangle$ vectors while mixing

$|p|_i 10 p|^i\rangle, |p|_i 01 p|^i\rangle$. Its matrix in that block is given by

$$[\Phi_i]_{2 \times 2} = \begin{pmatrix} \frac{\lambda_{z_i+1}}{\lambda_{z_i}} & \frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}} \\ \frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}} & \frac{\lambda_{z_i-1}}{\lambda_{z_i}} \end{pmatrix}. \quad (26)$$

It has two eigenvalues: 0 and $2 \cos \theta$, and consequently the eigenvalues of ρ_i in that block are A^{-1} and $-A^{-1}e^{-i2\theta}$ - the same eigenvalues of the $z_i = 1$ case. In fact, it is not hard to see that all the ρ_i operators are equivalent to each other.

The 2×2 matrix that transforms the standard basis elements $\{|p|_i 10, p|^i\rangle, |p|_i 01 p|^i\rangle\}$ to the eigenvectors basis is

$$V(z_i) \stackrel{\text{def}}{=} \frac{1}{\sqrt{\lambda_{z_i+1} + \lambda_{z_i-1}}} \begin{pmatrix} \sqrt{\lambda_{z_i+1}} & -\sqrt{\lambda_{z_i-1}} \\ \sqrt{\lambda_{z_i-1}} & \sqrt{\lambda_{z_i+1}} \end{pmatrix}. \quad (27)$$

Inside that subspace we have

$$V^\dagger(z_i) [\rho_i]_{2 \times 2} V(z_i) = A^{-1} \begin{pmatrix} -e^{-2\theta i} & 0 \\ 0 & 1 \end{pmatrix}. \quad (28)$$

The other vectors $\{|p|_i 00 p|^i\rangle, |p|_i 11 p|^i\rangle\}$, if exist, are trivial eigenvectors of ρ_i with eigenvalue A^{-1} .

Finally, we note that for $k > 5$, $H_{8,k,1}$ consists of exactly 14 paths² and hence it is a 14 dimensional space. These paths are shown graphically in Fig. 5. We identify each of them with a basis vector and label them by the numbers $1, \dots, 14$. Using this labeling we write down the block structure of the seven generators ρ_1, \dots, ρ_7 in Table 1. For each operator, the table lists the non-trivial blocks (i.e., blocks where Φ_i does not vanish). These are specified by brackets that surround the vectors that define them. The one-dimensional blocks correspond to the $z_i = 1$ case while the two-dimensional blocks correspond to the $z_i > 1$ case.

4.2 Proving the density

We will now prove the density part of Theorem 3.1. We will show that the seven operators ρ_i can approximate any special unitary matrix on $H_{8,k,1}$, provided that $k > 4$ and $k \neq 6, 10$. As it is a 14-dimensional space, we are interested in matrices $U \in SU(14)$ ³.

We begin by considering the action of ρ_1 and ρ_2 on this subspace. From Table 1 we see that these operators act non-trivially on the five 2×2 blocks $\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}, \{9, 12\}$, while applying the

²For $k = 5$ there are actually only 13 paths, as path 14 is illegal (it gets out of the graph). Nevertheless, it is easy to see that the density proof still holds in that border case. For $k = 4$ we cannot prove density (see theorem 4.1) while for $k < 4$ the $H_{8,k,1}$ is too small to encode 2 qubits.

³For $k = 5$ we look at $SU(13)$ and ignore the vector 14

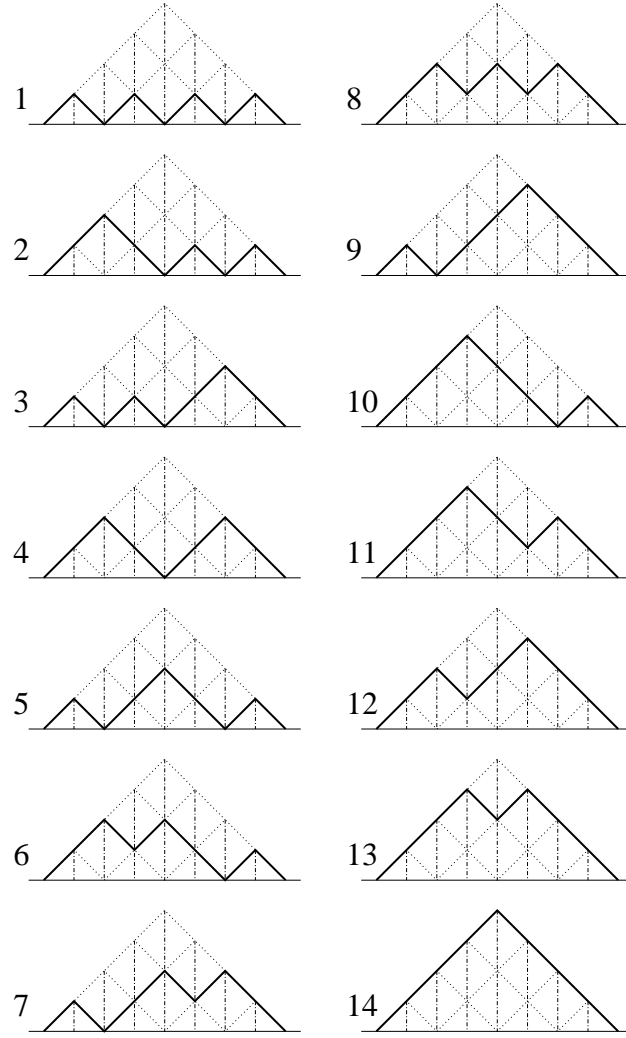


Figure 5: The 14 different vectors that correspond to paths on 8-strands, starting at 1 and ending at 1

| | | | | | |
|------------|--------|--------|---------|---------|----------|
| $\rho_1 :$ | (1) | (3) | (5) | (7) | (9) |
| $\rho_2 :$ | (1, 2) | (3, 4) | (5, 6) | (7, 8) | (9, 12) |
| $\rho_3 :$ | (1) | (3) | (6, 10) | (8, 11) | (12, 13) |
| $\rho_4 :$ | (1, 5) | (2, 6) | (3, 7) | (4, 8) | (13, 14) |
| $\rho_5 :$ | (1) | (2) | (7, 9) | (8, 12) | (11, 13) |
| $\rho_6 :$ | (1, 3) | (2, 4) | (5, 7) | (6, 8) | (10, 11) |
| $\rho_7 :$ | (1) | (2) | (5) | (6) | (10) |

Table 1: The block structure of the generators of B_8 in $H_{8,k,1}$ for $k > 5$.

trivial A^{-1} phase on the rest. In these blocks, the ρ_1 operator is represented by (i) , whereas the ρ_2 operator is represented by (i, j) . Additionally, the action of the operators on all five blocks is equivalent. The following theorem assures us that in each such block we may approximate any $SU(2)$ matrix.

Theorem 4.1 (Jones [12]) *If $k > 4$, and $k \neq 6, 10$, then in each 2×2 block, the group that is generated by ρ_1 and ρ_2 is dense in $SU(2)$.*

Proof: Given in appendix A.

Next, consider what happens when we are also allowed to act with ρ_3 . Looking at Table 1 we see that the resulting operators are diagonal on the blocks $\{1, 2\}$, $\{3, 4\}$, $\{5, 6, 10\}$, $\{7, 8, 11\}$, $\{9, 12, 13\}$. Obviously, we can still approximate any $SU(2)$ matrix in the 2×2 blocks. The next lemma provides a way to increase the dimensionality of the space on which we have density, in the following way: We start with $SU(A)$ and $SU(B)$, for two orthogonal subspaces A, B . If we also have a *bridge*, namely, some operator which mixes the two subspaces, then we have density in $SU(A \oplus B)$. This general lemma is very reminiscent of a lemma which appeared in an early version of [15]. Its proof uses a combination of ideas by Aharonov and Ben-Or [15] and from Kitaev [16].

Lemma 4.1 (The Bridge Lemma) *Consider a linear space C which is a direct sum of two subspaces A and B , and assume that $\dim B > \dim A \geq 1$. Then any $U \in SU(C)$ can be approximated to an arbitrary precision using a finite sequence of transformations from $SU(A)$, $SU(B)$ and any transformation $W \in SU(C)$ that mixes the two subspaces. Consequently, the group generated by $SU(A)$, $SU(B)$ and W is dense in $SU(C)$.*

Proof: Given in appendix A.

The bridge lemma implies that it is also possible to approximate any $SU(3)$ matrix in the 3×3 blocks. As an example, consider the $\{5, 6, 10\}$ block. From Theorem 4.1 we already know that we are able to approximate any $SU(2)$ transformation on the $\{5, 6\}$ block, and now we have the transformation ρ_3 that mixes this subspace with the one-dimensional subspace of that is spanned by tenth vector. Lemma 4.1 therefore guarantees that together they can approximate every transformation in $SU(3)$.

In the above reasoning there are two small cavities that are worth mentioning, since they will appear in the rest of the proof. Firstly, the mixing transformation ρ_3 does not belong to $SU(3)$. This, however, is not a real problem, as we can always consider the transformation $\tilde{\rho}_3 \stackrel{\text{def}}{=} c\rho_3$ with c some phase such fixes $\tilde{\rho}_3$ in $SU(3)$. Then $\langle \rho_1, \rho_2, \tilde{\rho}_3 \rangle$ is dense in $SU(3)$, and since $[SU(N), SU(N)] = SU(N)$ then also $[\langle \rho_1, \rho_2, \tilde{\rho}_3 \rangle, \langle \rho_1, \rho_2, \tilde{\rho}_3 \rangle]$ is dense in $SU(3)$. But the last group is equal to $[\langle \rho_1, \rho_2, \rho_3 \rangle, \langle \rho_1, \rho_2, \rho_3 \rangle]$ since the group bracket cancels out the phase c and therefore also $\langle \rho_1, \rho_2, \rho_3 \rangle$ is dense in $SU(3)$.

Secondly, we know we can *approximate* any transformation in $SU(2)$ while Lemma 4.1 assumes that we can get any transformation in $SU(2)$ precisely. But since the approximation is made of a *finite* product of operators, all of which can be approximated as accurately as desired by ρ_1, ρ_2, ρ_3 , it follows that we can also approximate any transformation in $SU(3)$ to any desired accuracy.

Naturally, the next step is to consider what happens when we are also allowed to act with ρ_4 . From Table 1 we see that resulting transformations will be invariant under the subspaces $\{1, 2, 5, 6, 10\}$, $\{3, 4, 7, 8, 11\}$, $\{9, 12, 13, 14\}$, that together make up the entire 14-dimensional subspace. We can use Lemma 4.1 again to learn that we can approximate any $SU(4)$ transformation in the $\{9, 12, 13, 14\}$ block. But what about the two other, five-dimensional blocks? There we cannot use Lemma 4.1 directly. To understand why this is so, consider, for example, the subspace $\{1, 2, 5, 6, 10\}$. We know that using ρ_1, ρ_2, ρ_3 we can approximate any $SU(2)$ transformation on the $\{1, 2\}$ block and any $SU(3)$ transformation on the $\{5, 6, 10\}$ block. We also know that that ρ_4 mixes these two blocks. However, to use Lemma 4.1 we must be able to approximate the $SU(2)$ transformations independently of the $SU(3)$ transformation. In other words, we must be able to approximate an $SU(2)$ transformation on the subspace $\{1, 2\}$, while leaving the subspace $\{5, 6, 10\}$ invariant and vice versa. But this is not a priori true since the transformations on $\{1, 2\}$ are generated by some sequence of the operators ρ_1, ρ_2, ρ_3 , which *simultaneously* generates some transformation on $\{5, 6, 10\}$. Luckily, we can use the fact that the dimensionality of the two subspaces is different in order to prove that such decoupling is possible:

Lemma 4.2 (The Decoupling Lemma) *Let G be an infinite discrete group, and let A, B be two finite linear spaces with different dimensionality. Let τ_a and τ_b be two homomorphisms of G into $SU(A)$ and $SU(B)$ respectively and assume that $\tau_a(G)$ is dense in $SU(A)$ and $\tau_b(G)$ is dense in $SU(B)$. Then for any $U \in SU(A)$ there exist a series $\{\sigma_n\}$ in G such that*

$$\tau_a(\sigma_n) \rightarrow U \quad (29)$$

$$\tau_b(\sigma_n) \rightarrow \mathbb{1} , \quad (30)$$

and vice versa.

Proof: Given in appendix A.

It is therefore clear that we are able to approximate any $SU(5)$ transformation on the $\{1, 2, 5, 6, 10\}$ and $\{3, 4, 6, 7, 8, 11\}$ blocks. Using ρ_5 we can now mix the $\{3, 4, 7, 8, 11\}$ subspace with the $\{9, 12, 13, 14\}$ subspace, and using the fact that their dimensionality is different, together with Lemmas 4.1, 4.2 - we are guaranteed that we can approximate any $SU(9)$ transformation on the combined 9-dimensional subspace.

Finally, by using ρ_6 , we mix the five-dimensional block $\{1, 2, 5, 6, 10\}$ with the nine-dimensional block from above - thereby approximating any transformation in $SU(14)$. This completes the density proof.

4.3 Proving the efficiency for the $k = \text{const}$ and $k = \text{poly}(n)$ cases

Having proved that the operators ρ_1, \dots, ρ_7 can approximate any matrix $U \in SU(14)$, the efficiency of the process for a *constant* k follows easily. We want to be able to approximate any $U \in SU(14)$ up to some error δ using $\text{poly}(1/\delta)$ operators in $\text{poly}(1/\delta)$ steps. This, and actually much more, can be achieved using the Solovay-Kitaev algorithm [16]. The algorithm relies on the fact that ρ_1, \dots, ρ_7 are dense in $SU(14)$, and provides us with a δ -approximation to U that consists of no more than $\text{poly}(\log(\delta^{-1}))$ operators in $\text{poly}(\log(\delta^{-1}))$ steps!

One should be aware, however, that the Solovay-Kitaev algorithm contains an initial step, where an ϵ -net is constructed. This is a finite set of operators that is generated by ρ_1, \dots, ρ_7 and has the property that every operator in $SU(14)$ is closer than ϵ to at least one of the elements of the net. ϵ is a finite constant which is unrelated to the target accuracy δ , and whose actual value is of the order 10^{-2} (see, for example, Ref [17]). The existence of such a net is guaranteed since we know that ρ_1, \dots, ρ_7 generate a dense set in $SU(14)$; its construction takes a constant time. We note that this constant may be very large since we might need to use brute force. Fortunately, the same ϵ -net can be used in all calculations, and we can treat its construction as a pre-computational stage.

The situation becomes more tricky when we let k be polynomially dependent on n . In such case k , and consequently the operators ρ_1, \dots, ρ_7 are no-longer constant, and a new ϵ -net has to be reconstructed for every n . Then we can no longer treat the ϵ -net construction as a constant step, and its complexity must be taken into account. The question is therefore whether we can still guarantee that the overall computational cost is polynomially bounded in k and in $\log(\delta^{-1})$? The answer is yes. The idea is to construct an ϵ -net for some k_0 and then use it to efficiently generate ϵ -nets for any large enough k .

We begin by considering the $k \rightarrow \infty$ limit of the ρ_i operators. From Sec. 4.1, we recall that in the standard basis these operators decompose into 1×1 or 2×2 blocks. The diagonalizing matrix of the 2×2 blocks is $V_k(z)$, given by Eq. (27). Here, and in what follows, we explicitly added the subscript k to $V(z)$ to indicate its dependence on k . In the limit $k \rightarrow \infty$, we have $\theta \rightarrow 0$ and we get

$$V_\infty(z) \stackrel{\text{def}}{=} \lim_{k \rightarrow \infty} V_k(z) = \frac{1}{\sqrt{2z}} \begin{pmatrix} \sqrt{z+1} & -\sqrt{z-1} \\ \sqrt{z-1} & \sqrt{z+1} \end{pmatrix}. \quad (31)$$

We now pick a finite k_0 - say, $k_0 = 7$ - and construct a new set of generators $\{\hat{\rho}_1, \hat{\rho}_2, \dots, \hat{\rho}_7\}$ by going through the 2×2 blocks and replacing each finite $V(z_i)$ by $V_\infty(z_i)$ while keeping the same eigenvalues. It is easy to see that also this set of operators generates a dense set in $SU(14)$. Indeed, the density proof in Sec. 4.2 remains valid since it only relies on the eigenvalues of the generating operators and on the fact that for $z > 1$, $V(z)$ mixes the two standard basis vectors. We will thus generate an ϵ -net from $\{\hat{\rho}_i\}$ and use it to generate the ϵ -net of $\{\rho_i\}$ for large k 's. The idea is simple: for large enough k , $V_k(z)$ is close

to $V_\infty(z)$, and at the same time the eigenvalues of the $\{\hat{\rho}_i\}$ are roughly $e^{2\pi i/k_0}$, whereas the eigenvalues of $\{\rho_i\}$ are roughly $e^{2\pi i/k}$. Therefore we can tentatively approximate every $\hat{\rho}_i$ by $\rho_i^{(k/k_0)}$, and obtain the desired ϵ -net. The exact formulation of this idea is contained in the following Lemma:

Lemma 4.3 *Let \hat{E} be an $\epsilon/2$ -net, generated from $\{\hat{\rho}_i\}$, and assume without loss of generality that each element in \hat{E} is a group commutator (this is possible since $SU(14)$ is a simple Lie-group and therefore $[SU(14), SU(14)] = SU(14)$). Then for large enough k , by replacing every occurrence of $\hat{\rho}_i$ in \hat{E} by $\rho_i^{2^m}$, with $m = \mathcal{O}(1/k)$, we obtain a new net, E_k , which is an ϵ -net.*

Proof:

Let d be the maximal number of generators that are needed to construct an element in \hat{E} . We wish to be able to approximate any $\hat{\rho}_i$ up to at least $\epsilon/2d$ using ρ_i . The first thing we take care of is that $V_k(z)$ will be close enough to $V_\infty(z)$. We therefore pick an integer K_1 such that for any $k > K_1$, $\|V(z) - V_\infty(z)\| \leq \epsilon/(6d)$.

Next, we must find a K_2 such that for any $k > K_2$, the eigenvalues of $\rho_i^{2^m}$ will be close enough to $\hat{\rho}_i$, for some yet to be determined m . This is more conveniently done by defining

$$P_i \stackrel{\text{def}}{=} A(k_0)\hat{\rho}_i, \quad Q_i \stackrel{\text{def}}{=} [A(k)\rho_i]^2, \quad (32)$$

and approximating the operators P_i with Q_i . In the end, the factors $A(k_0)$, $A(k)$ will cancel out when we plug these operators to the group commutator of each element in \hat{E} . The logic behind these definitions is that these factors cause one of the eigenvalues of both P_i and Q_i to be exactly one (see Eq. (10)), and therefore we only have to match the remaining eigenvalues. Indeed, the non-trivial eigenvalue of P_i is $e^{-i\pi(2+k_0)/k_0}$, whereas the non-trivial eigenvalue of Q_i is $e^{-4\pi i/k}$. We therefore define

$$m = \left\lfloor \frac{(2+k_0)/k_0}{4/k} \right\rfloor, \quad (33)$$

and let K_2 be such that for every $k > K_2$

$$\left| e^{-i\pi(2+k_0)/k_0} - e^{-4m\pi i/k} \right| < \epsilon/(6d). \quad (34)$$

It is easy to see that it is enough to take $K_2 > k_0$ for which $|e^{-4\pi i/K_2} - 1| < \epsilon/(6d)$.

Assume then that $k > \max(K_1, K_2)$ and let us estimate the distance between P_i and Q_i^m . This is the maximal distance between the corresponding blocks in the standard basis. In the 1×1 blocks both operators have an eigenvalue 1 and therefore the distance is zero. In the 2×2 blocks we have

$$[P]_{2 \times 2} = V_\infty^{-1}(z) \begin{pmatrix} e^{-i\pi(2+k_0)/k_0} & 0 \\ 0 & 1 \end{pmatrix} V_\infty(z), \quad (35)$$

$$[Q^m]_{2 \times 2} = V^{-1}(z) \begin{pmatrix} e^{-4m\pi i/k} & 0 \\ 0 & 1 \end{pmatrix} V(z), \quad (36)$$

and consequently

$$\begin{aligned} ||[P - Q^m]_{2 \times 2}|| &\leq ||V_\infty^{-1}(z) - V^{-1}(z)|| + \left| e^{-i\pi(2+k_0)/k_0} - e^{-4m\pi i/k} \right| \\ &+ ||V_\infty(z) - V(z)|| \leq \epsilon/(2d) . \end{aligned} \quad (37)$$

therefore $||P - Q^m|| < \epsilon/(2d)$.

Let us now return to the \hat{E} -net and create the E_k net. Any element in \hat{E} is a commutator of products of $\hat{\rho}_i$, and therefore remains unchanged if we replace $\hat{\rho}_i \rightarrow P_i$, because the phase factors cancel out in the commutator. The distance of this product from a product in which we replace $P_i \rightarrow Q_i^m$ is smaller than $\epsilon/2$ since $||P_i - Q_i^m|| < \epsilon/(2d)$ and we have at most d terms in the product. The Q_i^m 's product is unchanged upon the replacement $Q_i^m \rightarrow (\rho_i)^{2m}$ (again, the phase factors cancel out), thereby creating the E_k net. Hence any element in \hat{E} can be approximated up to a distance $\epsilon/2$ by an element of E_k . It follows that E_k is an ϵ -net. \square

It follows that we can create an ϵ -net from the operators ρ_i , and because $m < k$, the number of steps that are needed to create this net is bounded by $\text{poly}(k)$. The next step would be the application of the Solovay-Kitaev algorithm to approximate any transformation $U \in SU(14)$ up to an error δ - and so the overall computational cost is bounded by $\text{poly}(k, 1/\delta)$ as required.

A Proofs

A.1 Proof of Theorem 4.1

Proof:

Since ρ_1 and ρ_2 are not in $SU(2)$, we will look at their images under the canonical homomorphism $U(2) \rightarrow SU(2)$ which takes $V \in U(2)$ to $(\det V)^{-1/2}V$, and prove that these images form a dense set in $SU(2)$. Then using the fact that $[SU(2), SU(2)] = SU(2)$ it will follow that also ρ_1 and ρ_2 generate a dense set in $SU(2)$.

We first use the fact that group which is generated by ρ_1, ρ_2 is infinite as long as $k > 2$ and $k \neq 4, 6, 10$. This results was proved by Jones⁴ in 1983 and appears in Theorem 5.1 page 262 in ref [12]. It uses the canonical homomorphism between $SU(2)$ and $SO(3)$ and the well known classification of all finite subgroups in $SO(3)$.

To approximate any element in $SU(2)$ to within an ϵ , we pick two matrices in $g_1, g_2 \in G$ such that $||g_1 - g_2|| < \epsilon/3$ (we can do that since G has an infinite number of elements and $SU(2)$ is compact), and

⁴The theorem also shows that for $k = 10$, $\rho(B_n)$ is infinite provided that $n > 3$. This leads us to believe that the B_8 density and efficiency theorem is also valid for $k = 10$ - however this does not follow from our proof.

we define $g \stackrel{\text{def}}{=} g_1 g_2^{-1}$. Then obviously $\|g - \mathbb{1}\| < \epsilon/3$. Consequently, if $e^{\pm i\lambda}$ are the eigenvalues of g then $|e^{\pm i\lambda} - 1| < \epsilon/3$. Also g must be non-commuting with at least one of the matrices ρ_1 or ρ_2 which we shall denote by T .

Let U be the diagonalizing matrix of g : $g = U^{-1} \text{diag}\{e^{i\lambda}, e^{-i\lambda}\}U$, and define the two continuous families of matrices

$$R(\phi) \stackrel{\text{def}}{=} U^{-1} \text{diag}\{e^{i\phi}, e^{-i\phi}\}U, \quad (38)$$

$$S(\phi) \stackrel{\text{def}}{=} \sigma^{-1} R(\phi) \sigma. \quad (39)$$

Then it is easy to see that any matrix $V \in SU(2)$ can be presented as the product $R(\alpha)S(\beta)R(\gamma)$ for a suitable choice of $\alpha, \beta, \gamma \in \mathbb{R}$ (see, for example, Kitaev [16]). But since $|e^{i\lambda} - 1| < \epsilon/3$ then any member in the families $R(\cdot), S(\cdot)$ can be approximated by multiplications of g and σ up to a distance of $\epsilon/3$, and therefore the multiple $R(\alpha)S(\beta)R(\gamma)$ can be approximated to within ϵ . \square

A.2 Proof of Lemma 4.1 (The Bridge Lemma)

To prove lemma 4.1 we first need to prove the following two lemmas:

Lemma A.1 *Consider a linear space C which is a direct sum of two subspaces A and B such that $\dim B > \dim A \geq 1$, and let $W \in SU(C)$ be some transformation that mixes the two subspaces. Then for every pair of normalized vectors $|\psi\rangle, |\phi\rangle \in C$, we can approximate a transformation $T_{\psi \rightarrow \phi} \in SU(C)$ such that $T_{\psi \rightarrow \phi}|\psi\rangle = |\phi\rangle$, to any desired accuracy using a finite product of transformations from $SU(A)$, $SU(B)$ and W .*

Proof:

Instead of approximating the transformation $T_{\psi \rightarrow \phi}$ for any two vectors $|\psi\rangle, |\phi\rangle$, we will approximate a transformation T_ψ that transforms a particular vector $|v^*\rangle$ to an arbitrary vector $|\psi\rangle$. Then obviously, $T_{\psi \rightarrow \phi} = T_\psi T_\psi^{-1}$.

We begin by finding a vector $|v^*\rangle \in B$ for which $W|v^*\rangle \in B$. Such vector must exist since $\dim B > \dim A$. Indeed, let $|v_1\rangle, \dots, |v_n\rangle$ be a basis of B . Then $W|v_i\rangle = \alpha_i|u'_i\rangle + \beta_i|v'_i\rangle$, with $|u'_i\rangle \in A$ and $|v'_i\rangle \in B$. Then since $\dim B > \dim A$, the $|u'_i\rangle$ vectors are linearly dependent and we can find a non-trivial linear combination such that $\sum_i c_i \alpha_i |u'_i\rangle = 0$. Then the vector $|v^*\rangle \stackrel{\text{def}}{=} \sum c_i |v_i\rangle$ is in B and $W|v^*\rangle$ has no projection on A .

Next, we pick a vector $|u^*\rangle \in A$ for which $W|u^*\rangle$ has some projection on B . Such vector must exist because if $WA \subseteq A$ then by unitarity $WA = A$. This implies $WB = B$, again by unitarity - contradicting the assumption that W mixes the two subspaces.

Most generally, $W|u^*\rangle = a|u\rangle + b|v\rangle$ with $|u\rangle \in A$, $|v\rangle \in B$ and $|a|^2 + |b|^2 = 1$. If $a \neq 0$, we find a transformation $U \in SU(A)$ that takes $|u\rangle$ to $|u^*\rangle$, and define $\tilde{W} = UW$, otherwise, we set $\tilde{W} = W$. We have thus constructed a transformation \tilde{W} for which $\tilde{W}|v^*\rangle \in B$ and $\tilde{W}|u^*\rangle = a|u^*\rangle + b|v\rangle$ for some $0 \leq |a| < 1$.

Now let $|\psi\rangle = \alpha|u_0\rangle + \beta|v_0\rangle$ be an arbitrary vector, with $|u_0\rangle \in A$ and $|v_0\rangle \in B$. We will now apply a series of unitary operations that will take $|\psi\rangle$ closer and closer to $|v^*\rangle$. We start by moving $|u_0\rangle$ to $|u^*\rangle$ and $|v_0\rangle$ to $|v^*\rangle$ using transformations from $SU(A)$ and $SU(B)$ respectively, yielding the vector $|\psi_1\rangle = \alpha|u^*\rangle + \beta|v^*\rangle$. Using \tilde{W} we obtain

$$|\psi'_1\rangle = \tilde{W}|\psi_1\rangle = \alpha a|u^*\rangle + \alpha b|v\rangle + \tilde{W}|v^*\rangle. \quad (40)$$

As $|v\rangle, \tilde{W}|v^*\rangle \in B$, we can now apply a transformation from $SU(B)$ that takes $\alpha b|v\rangle + \tilde{W}|v^*\rangle$ to $c_2|v^*\rangle$. Here c_2 is the norm of $\alpha b|v\rangle + \tilde{W}|v^*\rangle$, which is usually smaller than one. We obtain

$$|\psi_2\rangle = \alpha a|u^*\rangle + c_2|v^*\rangle. \quad (41)$$

Notice that comparing $|\psi_2\rangle$ to $|\psi_1\rangle$, we see that we managed to move some of the weight from $|u^*\rangle$ to $|v^*\rangle$. We iterate this process. We get $|\psi'_2\rangle$ by applying the W_2 transformation on $|\psi_2\rangle$, and obtain $|\psi_3\rangle$ by moving the B part of $|\psi'_2\rangle$ to $|v^*\rangle$. After n such iterations we obtain

$$|\psi_n\rangle = \alpha a^{n-1}|u^*\rangle + c_n|v^*\rangle, \quad (42)$$

and since $|a| < 1$ it is obvious that we exponentially converge to $|v^*\rangle$, and in particular we can approximate T_ψ (and hence $T_{\psi \rightarrow \phi}$) to any desired accuracy using a finite number of transformations. \square

To continue, we need to be able to move a vector from subspace A to subspace B without affecting the rest of the vectors in subspace A . The following Lemma guarantees that this is possible.

Lemma A.2 *Under the same conditions of Lemma A.1, let $\{|u_1\rangle, \dots, |u_n\rangle\}$ be an orthonormal basis of A and $\{|v_1\rangle, \dots, |v_m\rangle\}$ be an orthonormal basis of B . Then using a finite product of transformations from $SU(A)$, $SU(B)$ and W , it is possible to approximate to any accuracy a transformation T that moves $|u_1\rangle$ to $|v_1\rangle$, while leaving the vectors $|u_2\rangle, \dots, |u_n\rangle$ unchanged.*

Proof: For $\dim A = 1$, the problem is trivial since we can simply use Lemma A.1. Assume then that $\dim A > 1$, and define the subspaces $A' \stackrel{\text{def}}{=} \text{span}\{|u_1\rangle \dots |u_{n-1}\rangle\}$ and $B' \stackrel{\text{def}}{=} \text{span}\{|v_2\rangle \dots |v_m\rangle\}$. By Lemma A.1 we can approximate a transformation \tilde{T} that takes $|u_n\rangle$ to $|v_1\rangle$. Consider now all the operators of the form $W = \tilde{T}^{-1}UV'\tilde{T}$ with $U \in SU(A)$ and $V' \in SU(B')$. Clearly, W takes $|u_n\rangle$ to itself - and therefore leaves invariant the subspace $A' \oplus B$. We claim that there is at least one such

transformation, $W^{(1)}$, that also mixes the subspaces A' and B . If this is indeed the case, then we can repeat the argument for the subspaces A' and B , which together with the particular transformation $W^{(1)}$ satisfy the conditions of Lemma A.1. Consequently, we find a transformation $W^{(2)}$ that takes $|u_{n-1}\rangle$ to $|v_1\rangle$ while leaving $|u_n\rangle$ unchanged and mixing the space that is spanned by $|u_1\rangle, \dots, |u_{n-2}\rangle$ with B . Repeating this again and again, we are left in the end with a transformation $W^{(n)}$ that transforms $|u_1\rangle$ to $|v_1\rangle$ and is the identity over $|u_2\rangle, \dots, |u_n\rangle$. Since this recursion has only n steps, it is clear that at any step we can approximate the mixing transformation $W^{(i)}$ to any desired accuracy using a finite product of operators from $SU(A)$, $SU(B)$ and W .

Let us now see why $W^{(1)}$ must exist. Indeed, if no such transformation exists, then for every two operators $U \in SU(A)$ and $V' \in SU(B')$ there is no mixing between the subspaces A' and B , and therefore there must exist operators $U' \in SU(A')$ and $V \in SU(B)$ such that

$$\tilde{T}^{-1}UV'\tilde{T} = U'V . \quad (43)$$

Then, for every $|u\rangle \in A$ and $|v\rangle \in B$,

$$\langle u|\tilde{T}^{-1}UV'\tilde{T}v\rangle = \langle u|U'Vv\rangle = 0 , \quad (44)$$

which implies that for every $U \in SU(A)$ and $V' \in SU(B')$,

$$\langle u\tilde{T}U|V'\tilde{T}v\rangle = 0 . \quad (45)$$

Notice that this equation holds also when we take one of the operators, U or V' , to be the identity. We will use this to show that $\tilde{T}A = A$ - in contradiction with the fact that $\tilde{T}|u_1\rangle = |v_1\rangle$.

We first deduce that $\tilde{T}A' \subset A$. To do this we show that for all $|u\rangle \in A'$, $\tilde{T}|u\rangle$ has no projection on B . We already know $\tilde{T}|u\rangle$ has zero projection on $|v_1\rangle$ (since $|u_1\rangle$ moves to $|v_1\rangle$), so it suffices to show that $\tilde{T}|u\rangle$ has no projection on B' .

Indeed, by Equation 45 it would suffice to show that $V'\tilde{T}v\rangle$ can be made to be an arbitrary vector in B' . This follows from the following reasoning. By the same argument as in the proof of Lemma A.1 there must be a vector $|v^*\rangle \in B$ such that $\tilde{T}|v^*\rangle \in B$. Moreover, $\tilde{T}|v^*\rangle$ must be in B' since

$$\langle v_1|\tilde{T}v^*\rangle = \langle v_1\tilde{T}^{-1}|v^*\rangle = \langle u_1|v^*\rangle = 0 . \quad (46)$$

Since $\tilde{T}|v^*\rangle \in B'$, using an arbitrary $V' \in SU(B')$, $V'\tilde{T}|v^*\rangle$ can be made to be any vector in B' .

Now pick any $|u^*\rangle \in A'$. Then $\tilde{T}|u^*\rangle \in A$, and therefore with an arbitrary transformation $U \in SU(A)$, $U\tilde{T}|u^*\rangle$ can be made to be an arbitrary vector in A . But since for every $|v\rangle \in B$ we have $\langle u^*\tilde{T}U|\tilde{T}v\rangle = 0$ then $\tilde{T}|v\rangle$ has no projection on A and consequently, $\tilde{T}B = B$. But then since \tilde{T} is unitary it follows that $\tilde{T}A = A$ - which is the contradiction we were seeking. \square

Having proved the last two lemmas, We are now in a position to prove Lemma 4.1:

Proof: Let $\{|u_1\rangle, \dots, |u_n\rangle\}$ be an orthonormal basis of A and similarly $\{|v_1\rangle, \dots, |v_n\rangle\}$ an orthonormal basis of B . We define the following sequence of subspaces

$$B_i = B_{i-1} \oplus |u_i\rangle \quad (47)$$

for $i = 1 \dots n$, where $B_0 = B$.

Let us show how to approximate an arbitrary $U \in SU(B_1)$, given $SU(B)$, $SU(A)$, and W . From Lemma A.2 we can use $SU(B)$, $SU(A)$ and W to approximate a transformation T that takes $|u_1\rangle$ to $|v_1\rangle$ while leaving the rest of the vectors in A intact. Therefore $T \in SU(B_1)$. Now pick an eigenvector $|\psi\rangle \in B_1$ of U with an eigenvalue $e^{i\theta}$. Using Lemma A.1 with respect to the subspaces $\text{span}|u_1\rangle, B$ and the mixing transformation T , we can approximate a transformation $W_\psi \in SU(B_1)$ that takes $|\psi\rangle$ to $|u_1\rangle$. We first show how to approximate the transformation $U_1 = W_\psi U W_\psi^{-1}$.

We notice that U_1 has $|u_1\rangle$ as an eigenvector with an eigenvalue $e^{i\theta}$. Consequently, U_1 leaves the subspace B invariant. Let V_1 be the transformation in $SU(B)$ that satisfies $V_1|v_1\rangle = e^{i\theta}|v_1\rangle$, $V_1|v_2\rangle = e^{-i\theta}|v_2\rangle$, and leaves the rest of the basis vectors unchanged. Recalling that T takes $|u_1\rangle$ to $|v_1\rangle$, we see that $T^{-1}V_1T$ has $|u_1\rangle$ as an eigenvector with eigenvalue $e^{i\theta}$ and leaves the subspace B invariant. So the only difference between U_1 and $T^{-1}V_1T$ is some transformation $V_2 \in SU(B)$, and therefore

$$U_1 = V_2 T^{-1} V_1 T, \quad (48)$$

and consequently,

$$U = W_\psi^{-1} V_2 T^{-1} V_1 T W_\psi. \quad (49)$$

Now that we have generated all transformations in $SU(B_1)$, we can generate $SU(B_2)$ using the very same procedure - except now B_1 plays the role of B and $|u_2\rangle$ plays the role of $|u_1\rangle$. In the same method we can work our way all up to $SU(B_n) = SU(C)$. \square

A.3 Proof of Lemma 4.2 (The Decoupling Lemma)

Proof: We define two subgroups $H_a \triangleleft SU(A)$ and $H_b \triangleleft SU(B)$ by

$$H_a \stackrel{\text{def}}{=} \left\{ U \in SU(A) \mid \exists \{\sigma_n\} \text{ s.t. } \begin{array}{l} \tau_a(\sigma_n) \rightarrow U \\ \tau_b(\sigma_n) \rightarrow \mathbb{1} \end{array} \right\}, \quad (50)$$

$$H_b \stackrel{\text{def}}{=} \left\{ V \in SU(B) \mid \exists \{\sigma_n\} \text{ s.t. } \begin{array}{l} \tau_a(\sigma_n) \rightarrow \mathbb{1} \\ \tau_b(\sigma_n) \rightarrow V \end{array} \right\}. \quad (51)$$

The theorem will be proved once we show that $H_a = SU(A)$ and $H_b = SU(B)$.

To do that, we first observe that both H_a and H_b are normal subgroups. It is also straightforward to see that they are closed. Consider, for example, H_a in $SU(A)$: assume that $\{U_k\}$ in H_a converges to $U \in SU(A)$. Then there exist series $\sigma_n^{(k)}$ such that

$$\lim_{n \rightarrow \infty} \tau_a(\sigma_n^{(k)}) = U_k , \quad (52)$$

$$\lim_{n \rightarrow \infty} \tau_b(\sigma_n^{(k)}) = \mathbb{1} . \quad (53)$$

Without loss of generality, we may choose the series such that for every n, k ,

$$\|\tau_b(\sigma_n^k) - U_k\| < 1/n \quad , \quad \|\tau_b(\sigma_n^k) - \mathbb{1}\| < 1/n . \quad (54)$$

Then the series $\tau_a(\sigma_k^k) \rightarrow U$, and we are guaranteed that $\tau_b(\sigma_k^k) \rightarrow \mathbb{1}$.

Now, any non-trivial normal subgroup of $SU(N)$ must be finite⁵. Therefore if we show that H_a and H_b are infinite it will follow that $H_a = SU(A)$ and $H_b = SU(B)$. To do that, we first show that there is an isomorphism of groups M (a 1-1 and onto mapping which preserves the action of the group) between the coset groups $SU(A)/H_a$ and $SU(B)/H_b$. We define M as follows: $M(UH_a) = VH_b$ if there exists a series $\{\sigma_n\}$ such that

$$\tau_a(\sigma_n) \rightarrow U \quad , \quad \tau_b(\sigma_n) \rightarrow V . \quad (55)$$

We first need to show that this function is well defined for all cosets UH_a . This follows from:

- For each coset UH_a , there exists at least one series that satisfies the requirements of the definition of M . Indeed, pick a series $\{\sigma_n\}$ such that $\tau_a(\sigma_n) \rightarrow U$. Then the series $\tau_b(\sigma_n)$ in $SU(B)$ must have a limiting point since $SU(B)$ is compact. Therefore there exists a sub-series $\{\sigma_{n_k}\}$ such that $\tau_a(\sigma_{n_k}) \rightarrow U$ and $\tau_b(\sigma_{n_k}) \rightarrow V$. We have $M(UH_a) = VH_b$
- $M(UH_a)$ is defined uniquely. Indeed, assume there exist two series $\{\sigma_n^{(1)}\}$ and $\{\sigma_n^{(2)}\}$ such that

$$\tau_a(\sigma_n^{(1)}) \rightarrow U_1 \quad , \quad \tau_b(\sigma_n^{(1)}) \rightarrow V_1 , \quad (56)$$

$$\tau_a(\sigma_n^{(2)}) \rightarrow U_2 \quad , \quad \tau_b(\sigma_n^{(2)}) \rightarrow V_2 , \quad (57)$$

with U_1 and U_2 in the coset UH_a . We will show that V_1 and V_2 must be in the same coset of H_b . Denote $\Delta \stackrel{\text{def}}{=} U_1 U_2^{-1}$. Since H_a is normal, Δ is in H_a and we may therefore find a series $\{\sigma_n^{(3)}\}$ such that

$$\tau_a(\sigma_n^{(3)}) \rightarrow U_1 U_2^{-1} \quad , \quad \tau_b(\sigma_n^{(3)}) \rightarrow \mathbb{1} . \quad (58)$$

⁵This follows from the fact that the quotient group $SU(N)/Z(SU(N))$ is a simple group, and $Z(SU(N))$ - the center of $SU(N)$ - is finite. See for example Theorem 11.26, at page 108 of Ref [18]

Then looking at the series $\sigma_n^{(4)} \stackrel{\text{def}}{=} (\sigma_n^{(1)})^{-1} \sigma_n^{(3)} \sigma_n^{(2)}$, we find that

$$\tau_a(\sigma_n^{(4)}) \rightarrow U_1^{-1} U_1 U_2^{-1} U_2 = \mathbb{1} \ , \quad \tau_b(\sigma_n^{(4)}) \rightarrow V_1^{-1} V_2 \ . \quad (59)$$

Therefore $V_1^{-1} V_2 \in H_b$, and so $V_1 H_b = V_2 H_b$.

It remains to show that M is $1-1$, onto, and preserves the action of the group. The onto part follows if we start with a series that converges to some V in VH_b and apply the same reasoning as in the first item above. The $1-1$ part follows if we apply the same reasoning as in the second point above, starting with the V_1, V_2 in the same coset instead of the opposite direction. The fact that M is a homomorphism follows from the fact that τ is a representation.

Recall now that H_a and H_b can be either finite groups or equal to their “supergroup”. So there are four possibilities:

1. H_a is finite and $H_b = SU(B)$.
2. H_b is finite and $H_a = SU(A)$.
3. Both H_a and H_b are finite.
4. $H_a = SU(A)$ and $H_b = SU(B)$.

The first and second cases are impossible since, for example, if H_a is finite and $H_b = SU(B)$ then $SU(B)/H_b$ has only one coset while $SU(A)/H_a$ has infinitely many - and thus they cannot be related by a $1-1$ onto map.

Let us now see why the third case is also impossible. To do this, we will show that M is a continuous map. Indeed, assume that $U_k H_a \rightarrow U H_a$ (here, convergence means that for some representatives of the cosets we have $U_k \rightarrow U$. It is easy to check that this is well defined). Then let $M(U_k H_a) = V_k H_b$, $M(U H_a) = V H_b$. We will show that $V_k H_b \rightarrow V H_b$. The proof is straightforward, similarly to the proof that H_a is closed. pick a series of series $\{\sigma_n^{(k)}\}$ such that

$$\lim_{n \rightarrow \infty} \tau_a(\sigma_n^{(k)}) = U_k \ , \quad (60)$$

$$\lim_{n \rightarrow \infty} \tau_b(\sigma_n^{(k)}) = V_k \ , \quad (61)$$

and without any loss of generality we assume that

$$\|\tau_a(\sigma_n^k) - U_k\| < 1/n \ , \quad \|\tau_b(\sigma_n^k) - V_k\| < 1/n \ . \quad (62)$$

Then since $U_k \rightarrow U$, we have $\tau_a(\sigma_k^k) \rightarrow U$, and since $M(U H_a) = V H_b$ we can find a sub-series $\tau_b(\sigma_{k_\ell}^{k_\ell})$ that converges to some $\tilde{V} \in V H_b$. In order not to overload the notation, let us re-define k to be that

sub-series. We now claim $V_k \rightarrow \tilde{V}$. Indeed, for each $\epsilon > 0$, we can choose K such that for each $k > K$, $1/k < \epsilon/2$ and $\|\tau_b(\sigma_k^k) - \tilde{V}\| < \epsilon/2$. Then

$$\|V_k - \tilde{V}\| \leq \|V_k - \tau_b(\sigma_{n_k}^k)\| + \|\tau_b(\sigma_{n_k}^k) - \tilde{V}\| \leq \epsilon. \quad (63)$$

Now H_a and H_b are closed normal subgroups, and therefore $SU(A)/H_a$ and $SU(B)/H_b$ are Lie groups themselves (see for example, Theorem 3.64, pp 124, in [19]). Furthermore, since every continuous homomorphism between Lie groups is also smooth (see for example, Theorem 3.39, pp 109, in [19]), we have found a smooth diffeomorphism (1 – 1 homeomorphism) between two differentiable manifolds. However, since both H_a and H_b are finite then

$$\dim SU(A)/H_a = \dim SU(A) \neq \dim SU(B) = \dim SU(B)/H_b, \quad (64)$$

and it is therefore impossible to find a diffeomorphism between the two manifolds. \square

References

- [1] Freedman M. H., Kitaev A., Wang Z. *Simulation of topological field theories by quantum computers* Commun. Math. Phys. 227 (2002) 587-603
- [2] Freedman M. H., Larsen M., Wang Z. “A Modular Functor which is Universal for Quantum Computation”, Commun. Math. Phys., 2002, 227, pp 605-622.
- [3] Aharonov D., Jones V. F, Landau Z., In proceedings of the 38th ACM Symposium on Theory of Computing (STOC 2006) Seattle, Washington, USA. [arxiv:quant-ph/0511096](https://arxiv.org/abs/quant-ph/0511096)
- [4] Jones V. F. R, “A polynomial invariant for via Von Neumann algebras”, Bull. Amer. Math. Soc (N. S.), 1985, 12, pp 103-111
- [5] Witten E., “Quantum Field Theory and the Jones Polynomial”, Commun. Math. Phys., (1989), 121, pp 351-399.
- [6] Freedman M. H., *P/NP and the quantum field computer*, Proc. Natl. Acad. Sci., USA, **95**, (1998), 98–101
- [7] Freedman M. H., Kitaev A., Larsen M., Wang Z., *Topological quantum computation. Mathematical challenges of the 21st century* (Los Angeles, CA, 2000). Bull. Amer. Math. Soc. (N.S.) 40 (2003), no. 1, 31–38

- [8] Bordewich M., Freedman M., Lovasz L., Welsh D., " *Approximate counting and quantum computation* ", to appear in Combinatorics, Probability and Computing, 2006.
- [9] Kitaev A. Yu., private communication, 2005
- [10] Wocjan P., Yard J. "The Jones polynomial: quantum algorithms and applications in quantum complexity theory" [arxiv:quant-ph/0603069](#)
- [11] Garnerone S., Marzuoli A., Rasetti M., " *Quantum automata, braid group and link polynomials* ", 2006, [arxiv:quant-ph/0601169](#)
- [12] Jones V. F. R., "Braid groups, Hecke algebras and type II_1 factors". In: Geometric methods in operator algebras, Proc. of the US-Japan Seminar, Kyoto, July 1983. See Theorem 5.1, page 262.
- [13] Artin E., Theorie der Zöpfe, Hamburg Abh. , (1925), 4, pp 47-72
- [14] Kauffman L., "State models and the Jones polynomial", Topology, 26, (1987), 395-407.
- [15] D. Aharonov, M. Ben-Or., " *Fault-tolerant quantum computation with constant error* ", Proc. ACM STOC, pp. 176–188, (1997). [arxiv:quant-ph/9906129](#)
- [16] Kitaev A. Yu., Shen A. H., Vyalyi M. N , "Classical and quantum computation", vol 47 of *Graduate Studies in Mathematics*. Amsterdam Mathematical Society, Providence, Rhode Island, 2002.
- [17] C. M. Dawson, M. A. Nielsen, " *The Solovay-Kitaev algorithm* ", [arxiv:quant-ph/0505030](#)
- [18] L. C. Grove, " *Classical Groups and Geometric Algebra* , Graduate Studies in Mathematics; V 39, American Mathematical Society, 2001
- [19] Warner F. W., "Foundations of Differentiable Manifolds and Lie Groups", Springer, (1983) (Second edition).

